8

Executing VBA

In the old days of programming, *procedural* languages ruled, meaning that the overall program execution traveled from top to bottom. The main body of any of these programs had to cover every possibility: display a screen to the user, gather input, perform edit checking, display messages, update the database (or simple files in those days), and close when everything was done. The main program also had to cover every option or side request that the user might make. This made it difficult to understand the entire program, and it was tough to make changes because everything had to be retested when a modification was made. These lumbering beasts included COBOL, RPG, Pascal, and earlier forms of Basic. Millions of lines of code were written in these languages.

Fortunately, those days are over for VBA programmers. VBA is an *event-driven* language. In every Access form and report there are a variety of events that are waiting for you to use. They are available when the form opens and closes, when records are updated, even when individual fields on the screen are changed. It's all there at your fingertips. Each event can contain a procedure, which is finally where we get back to our procedural roots. Although each procedure runs from top to bottom, just like in the old days, it only runs when the event *fires*. Until then, it sleeps quietly, not complicating your logic or slowing down your program.

Event-driven programming makes it much easier to handle complex programming tasks. By only worrying about events in your coding when they actually happen, each procedure is simpler and easier to debug.

In this chapter, we'll explore the nature of VBA events and show how the most common events are used, and we'll look at how two different sections of your VBA code can run at the same time. We'll also provide some guidelines about when and how to use Public and Private procedures, class modules, and data types. Finally, we'll outline structural guidelines for procedures, show some common string and date handling techniques, and also how to prevent rounding errors in your calculations.